Mysteries of the Deep: What happens inside of MPI on Blue Gene/Q and why it matters

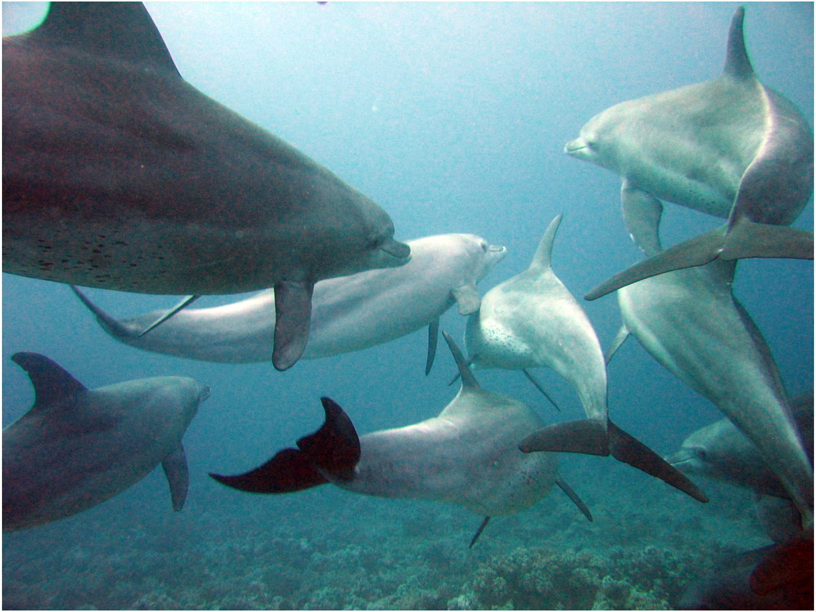
Jeff Hammond

Leadership Computing Facility
Argonne National Laboratory

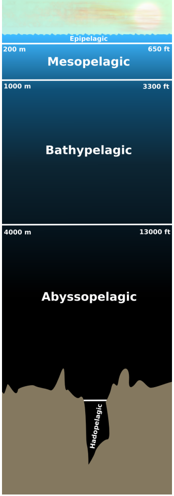
March 5, 2013

The view from the boat

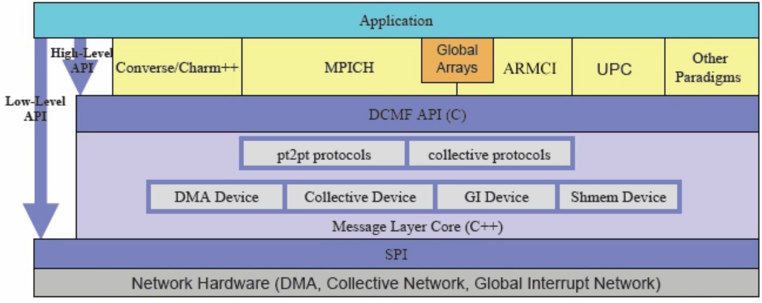
A reason to dive



But not too deep

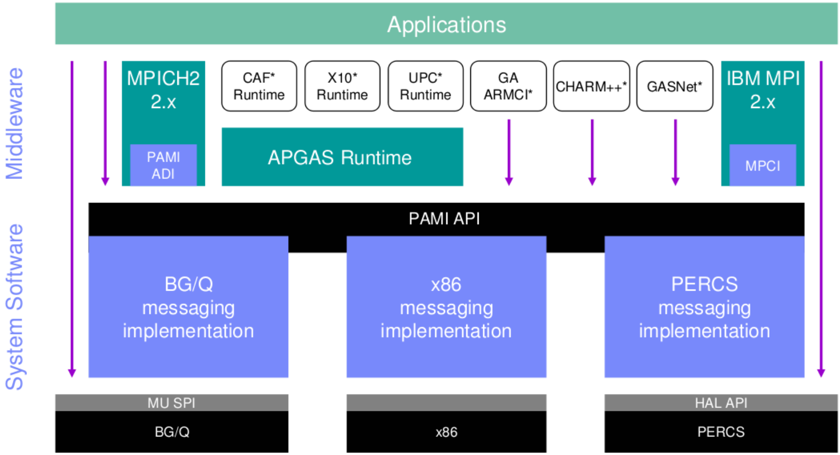


Blue Gene/P Communication architecture



Source: IBM

Blue Gene/Q Communication architecture



*all runtimes are not supported on all platforms

MPI-3 on Blue Gene/Q

There is no official plan to support MPI-3 on Blue Gene/Q but a subset of these features may be available at some point in the future.

I have build MPICH-3.0.x on Blue Gene/Q and many things pass while others are completely broken, but this is not a supported usage and you should not do it unless you are `l33t h4xz0r`.

MPI-3 on Blue Gene/Q

A portable implementation of MPI-3 nonblocking collectives (NBC) on top of MPI-1 is provided by LibNBC.

NBC are mostly implemented in BG/Q MPI source already, but they don't work yet.

It is debatable if NBC has any value on BG/Q because BC are so fast. You will need to enable asynchronous progress (which consumes hw threads) and have enough computation to hide 10-100 times the BC latency.

MPI-3 on Blue Gene/Q

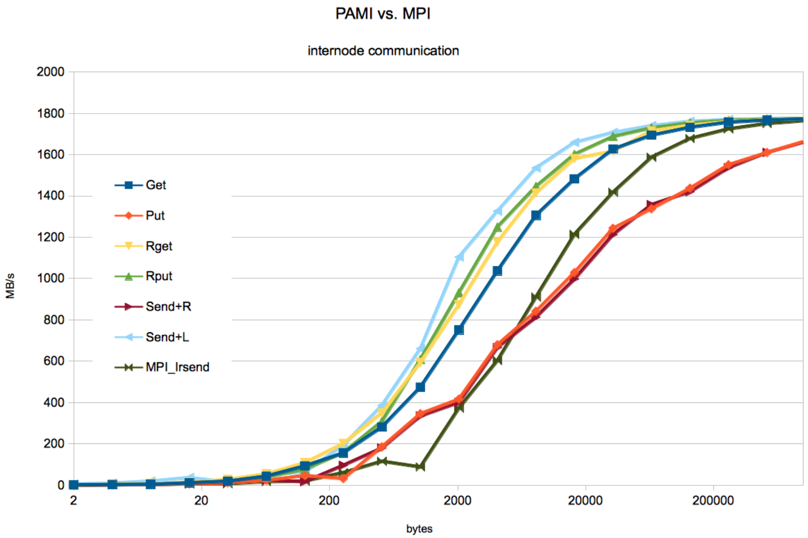
All of the features of MPI-3 NBC and RMA are available already from PAMI. *I will show how to implement MPI-3 RMA features using PAMI on Thursday.*

Many other features of MPI-3 can be implemented on Blue Gene/Q using a mixture of MPI, MPIX, PAMI and SPI calls. If you are using MPI-3 on other systems and would like these features on Blue Gene/Q, please talk to me offline.

If you are point-to-point communication limited, using PAMI will really help performance but not be portable.

(I can show you how to write the equivalent to PAMI on Cray XE/XK/XC though...)

MPI versus PAMI



Performance results

Neighbor exchange

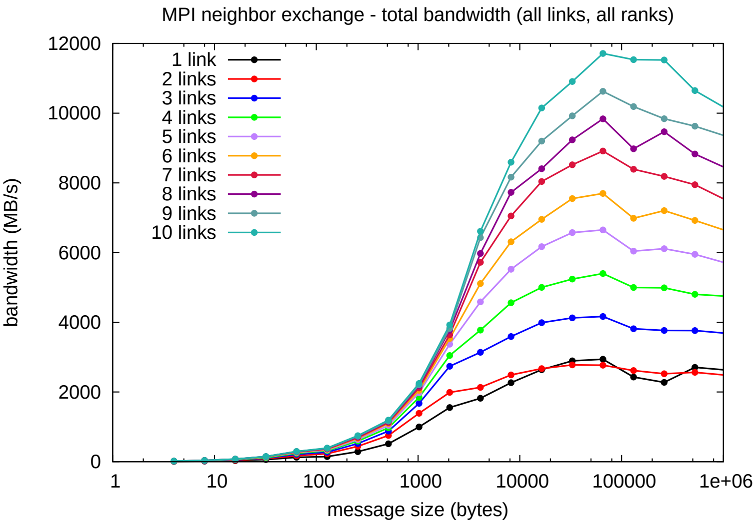
Testing injection (send) bandwidth along 1 to 10 links.

Explicitly mapped to torus.

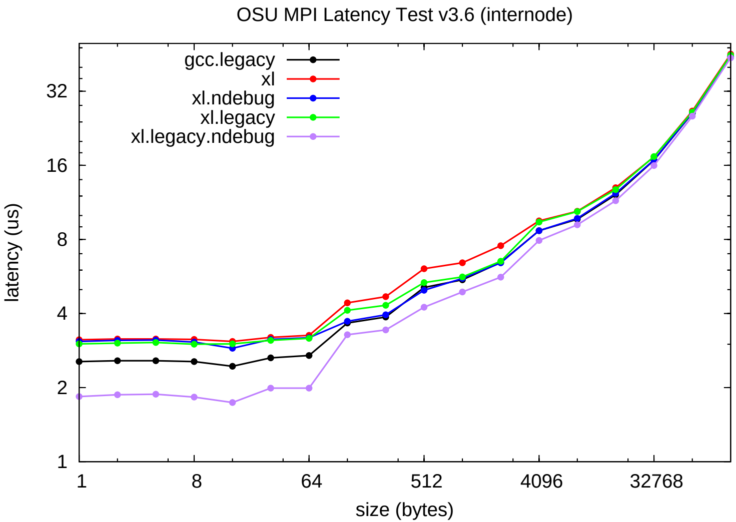
Nonblocking recv and send followed by waitall.

No repetition in test but warmup along all 6 links done first.

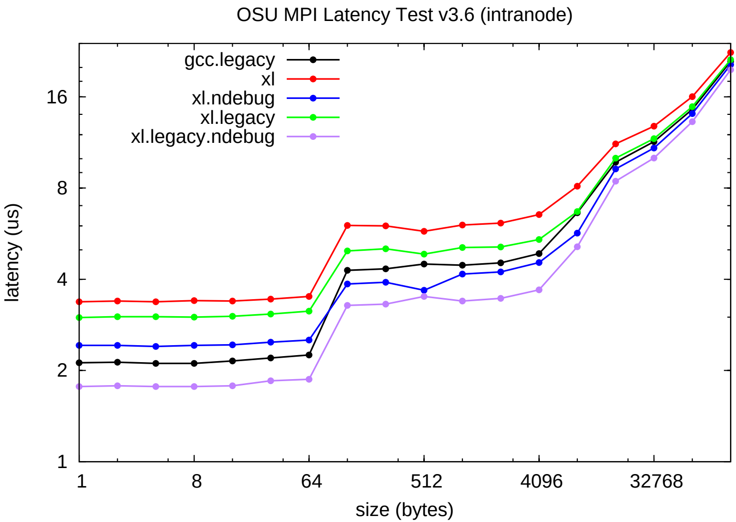
Neighbor exchange



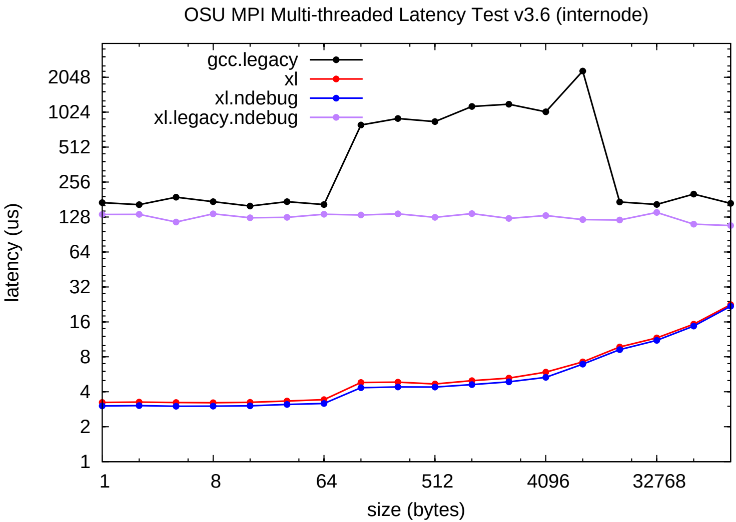
OSU MPI Benchmarks



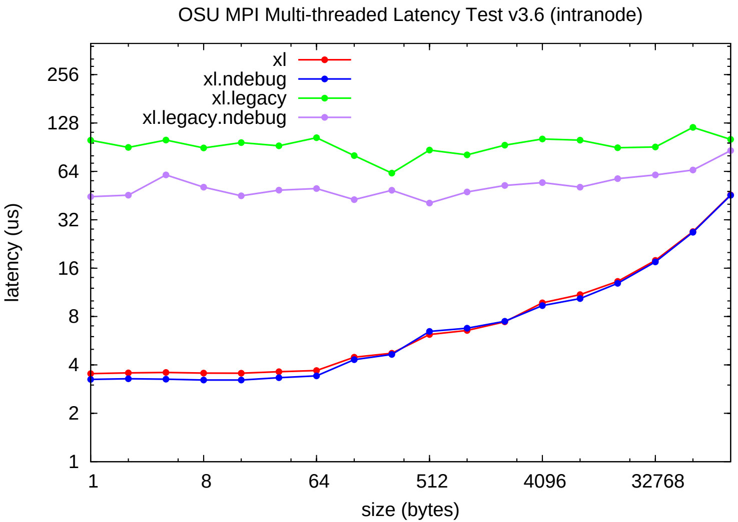
OSU MPI Benchmarks



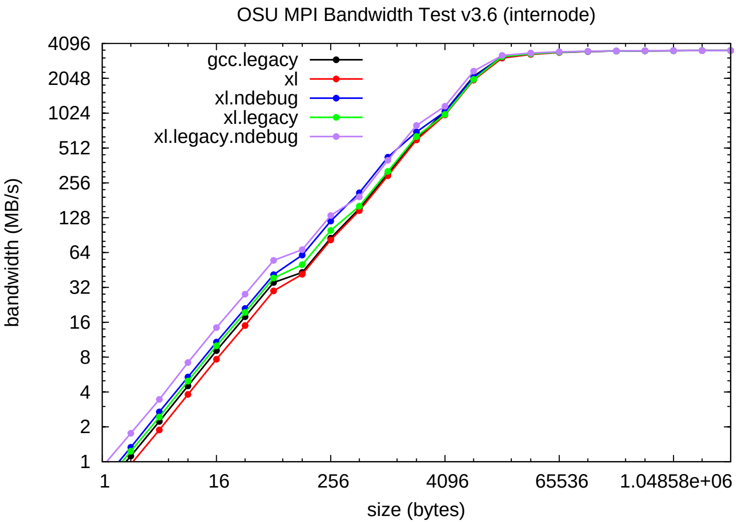
OSU MPI Benchmarks



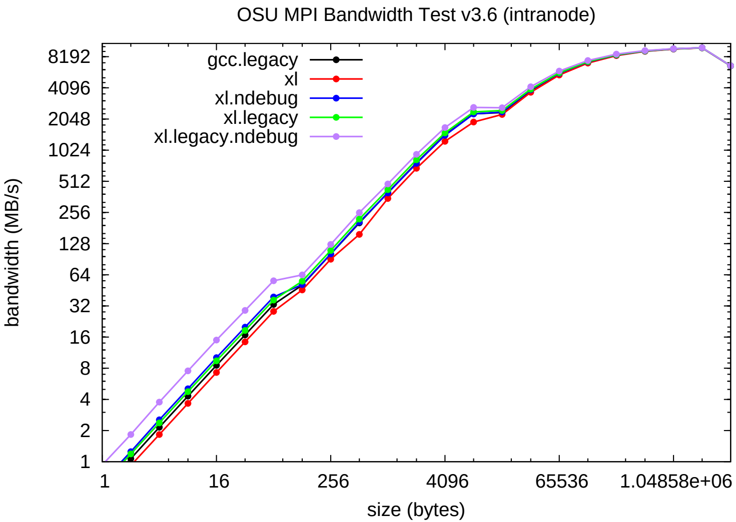
OSU MPI Benchmarks



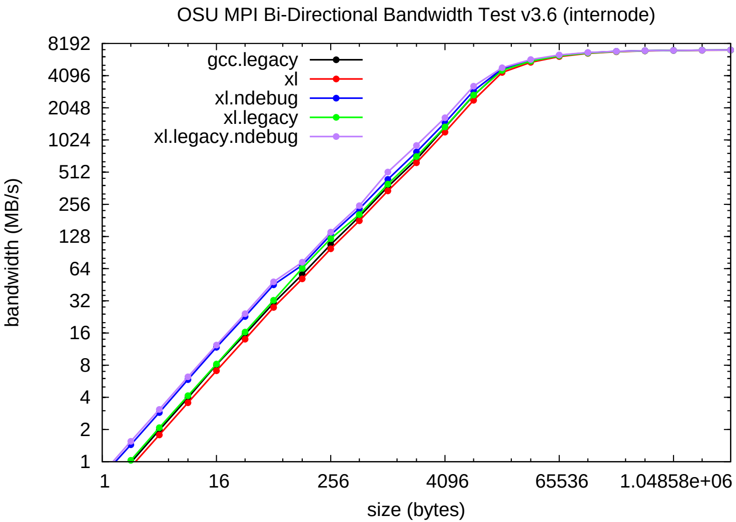
OSU MPI Benchmarks



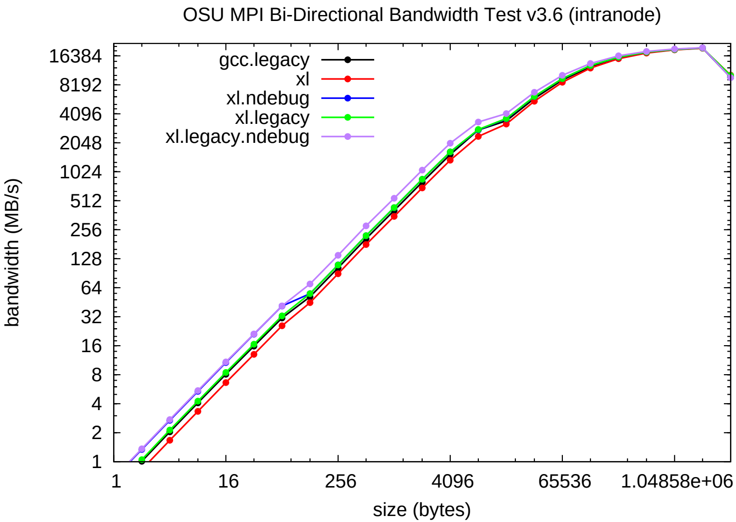
OSU MPI Benchmarks



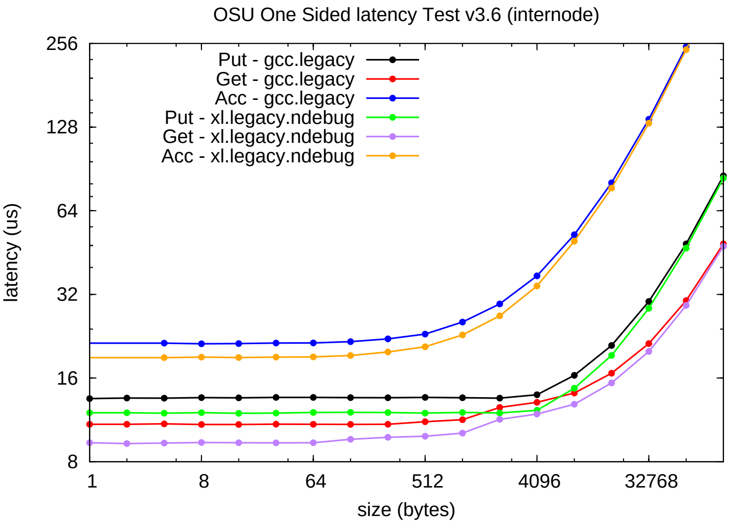
OSU MPI Benchmarks



OSU MPI Benchmarks



OSU MPI Benchmarks



Topology and MPIX

Topology API

From /bgsys/drivers/ppcfloor/comm/*/include/mpix.h

```
#define MPIX_TORUS_MAX_DIMS 5
typedef struct
{
    unsigned ppn;
    unsigned coreID;
    unsigned torus_dimension;
    unsigned Size[MPIX_TORUS_MAX_DIMS];
    unsigned Coords[MPIX_TORUS_MAX_DIMS];
    unsigned isTorus[MPIX_TORUS_MAX_DIMS];
} MPIX_Hardware_t;
```

Topology API

```
MPIX_Hardware_t hw;
MPIX_Hardware(&hw);

int hopCoord = (hw.Coords[0]+1) % (hw.Size[0]);
int tempCoords[MPIX_TORUS_MAX_DIMS+1] =
    { hopCoord, hw.Coords[1], hw.Coords[2],
       hw.Coords[3], hw.Coords[4], hw.coreID};
MPIX_Torus2rank(tempCoords, &rank_ap);
```

Environment variables

PAMI Verbosity

PAMID_STATISTICS Turns on statistics printing for the message layer such as the maximum receive queue depth. *Disabled by default.*

PAMID_VERBOSE Always set this to 1, which will add only 20 lines to the top of your output file, but contains extremely valuable information.

Setting this to 2 or 3 provides substantial output that is useful for debugging performance, particularly of collectives.

Disabled by default.

PAMI High-level tuning options

PAMID_THREAD_MULTIPLE

Enabled by default with **MPI_THREAD_MULTIPLE** when using non-`legacy` MPI libraries.

Use this when you need one or more of the following:
(1) asynchronous progress on send-recv,
(2) one-sided communication,
(3) massive injection rate e.g. MMPS, SQMR, GUPS benchmarks.

Optimizing collectives through thinking

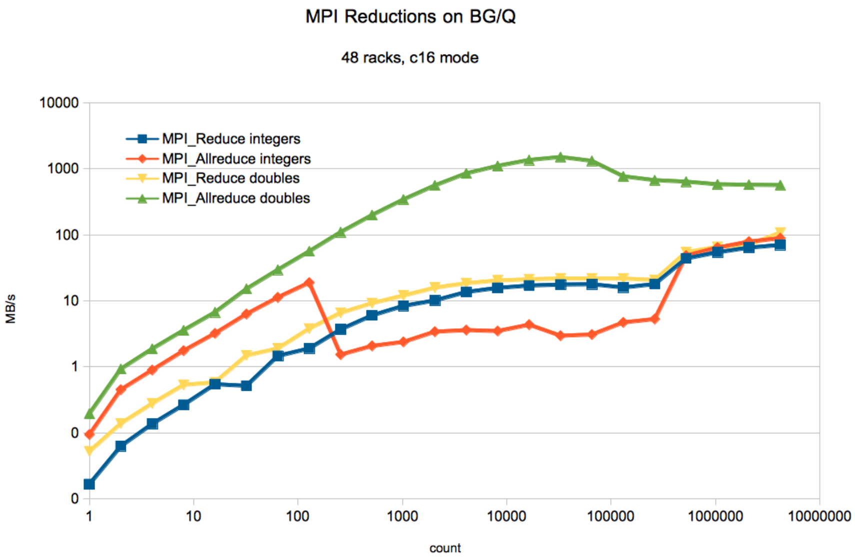
MPI_Reduce_scatter: reduce a buffer to root, then scatter from root. This MPI-2.1 function requires vector arguments, so this is really reduce, then scatterv. MPI_Scatterv is not optimized. In addition, the arguments to scatterv must be allocated internally. At scale, this consumes a lot of memory (perhaps as much as 8x).

MPI_Reduce_scatter_block: reduce a buffer to root, then scatter from root. This MPI-2.2 function takes scalar arguments and therefore uses less memory at scale.

MPI_Allreduce: reduce a buffer everywhere, then copy out my portion. This MPI-1 function requires scalar arguments and is highly optimized on Blue Gene/P. Using MPI_IN_PLACE means no extra memory is used (although the input buffer is modified).

Replace **MPI_Alltoallv** with **MPI_Alltoall** if you padding is easy; for sparse **MPI_Alltoallv**, you may be better off using point-to-point.

MPI Reduction Surprises



Closing thoughts

If you know your algorithms are good and communication is holding you back, understanding the internals of MPI will help you write faster and more scalable code on Blue Gene.

P.S. If profiling shows you spend too much time in MPI_Barrier, your *algorithm* is the problem (load imbalance), not communication.

See `https://wiki.alcf.anl.gov/parts/index.php/···`
`···Mira_MPI_Documentation` for more information (this will soon migrate to ALCF docs).

Please email me if you have questions: `e@anl.gov`.